NASA Langley's Research and Technology-Transfer Program in Formal Methods

Ricky W. Butler
Victor A. Carreño
Ben L. Di Vito
Kelly J. Hayhurst
C. Michael Holloway
Paul S. Miner
Gerald Lüttgen (ICASE)
César Muñoz (ICASE)

Assessment Technology Branch
NASA Langley Research Center
Hampton, Virginia

http://shemesh.larc.nasa.gov/fm.html

June 2000

Abstract

This paper presents an overview of NASA Langley's research program in formal methods. The major goals of this work are to make formal methods practical for use on high integrity systems, to orchestrate the transfer of this technology to U.S. industry through use of carefully designed demonstration projects, and to exploit this technology to help achieve NASA's goals in aeronautics. Several direct technology transfer efforts have been initiated that apply formal methods to critical subsystems of real aerospace computer systems.

Contents

1 Rationale For a Formal Methods Research Program

NASA Langley Research Center has been developing techniques for the design and validation of flight critical systems for over two decades. Although much progress has been made in developing methods to accommodate physical failures, design flaws remain a serious problem [73, 94, 49, 1, 65, 43, 106].

The following recent events show the potential of design errors for disaster:

- The maiden flight of the Ariane 5 launcher (June 4 1996) ended in an explosion. Total loss was over $850 million.
- Between June 1985 and January 1987, a computer-controlled radiation therapy machine, called the Therac-25, massively overdosed six people, killing two.
- Replacement of defective Pentium processors costs Intel Corp. $500 million in 1995.
- The April 30, 1999 loss of a Titan I, which cost the taxpayers $1.23-billion, was due to incorrect software (incorrectly entered roll rate filter constant)
- December 1999 loss of the Mars Polar Lander was due to an incomplete software requirement. A landing leg jolt caused engine shutdown.
- Denver Airport's computerized baggage handling system delayed opening by 16 months. Airport cost was $3.2 billion over budget.
- Patriot failure at Dharan (software error put tracking off by 0.34 of a second)

A recent (January 24, 1999) report from the Office of Science and Technology Policy entitled Information Technology For The Twenty-First Century: A Bold Investment In America's Future states

> Software research was judged by The President's Information Technology Advisory Committee to be the highest priority area for fundamental research. From the desktop computer to the phone system to the stock market, our economy and society have become increasingly reliant on software. This Committee concluded that not only is the demand for software exceeding our ability to produce it; the software that is produced today is fragile, unreliable, and difficult to design, test, maintain, and upgrade.

Although the aviation industry has been more conservative and cautious in its adoption of information technology than most other industries, it is beginning to suffer from the effects of software. David W. Robb (editor) writes in the Oct 1996 issue Avionics Magazine:

> Avionics have never been more clearly at center stage. The benefits of flat-panel and heads-up displays, the precision of GPS positioning, ... and the flexibility of integrated avionics, to name just a few areas, are transforming aviation almost faster than we can print these words.
>
> It is no secret that aircraft are becoming ever more dependent on their onboard electronics. The emerging world of CNS and Free Flight promises to accelerate this trend dramatically. As the equipment grows more capable and sophisticated, so does the challenge of testing and maintaining it.

Harry C. Stonecipher, President and Chief Operating Officer The Boeing Company. wrote in an article entitled "Getting It Right: Defense Acquisition for the 21st Century" May 26, 1999:

> ... avionics systems account for about one-third of the fly-away cost of a military aircraft and a significant amount of its life-cycle cost. It goes without saying that our warfighters are increasingly dependent upon the use of avionics systems for everything from navigation to targeting and to battlefield management.

1.1 The Problem With Software and Hardware

Digital systems (both hardware and software) are notorious for their unpredictable and unreliable behavior:

> Studies have shown that for every six new large-scale software systems that are put into operation, two others are cancelled. The average software development project overshoots its schedule by half; larger projects generally do worse. And three quarters of all large systems are "operating failures" that either do not function as intended or are not used at all.
>
> Despite 50 years of progress, the software industry remains years–perhaps decades–short of the mature engineering discipline needed to meet the demands of an information-age society[44].

Lauren Ruth Wiener describes the software problem in her book, *Digital Woes: Why We Should Not Depend Upon Software*:

> Software products—even programs of modest size—are among the most complex artifacts that humans produce, and software development projects are among our most complex undertakings. They soak up however much time or money, however many people we throw at them.
>
> The results are only modestly reliable. Even after the most thorough and rigorous testing some bugs remain. We can never test all threads through the system with all possible inputs[134].

The hardware industry also faces serious difficulties, as evidenced by the 1994 design error in the Pentium floating-point unit. In response to an outcry over the design flaw in the Pentium floating point unit, Intel's President, Andy Grove, wrote on the comp.sys.intel Internet bulletin board:

> After almost 25 years in the microprocessor business, I have come to the conclusion that no microprocessor is ever perfect; they just come closer to perfection with each stepping. In the life of a typical microprocessor, we go thru [sic] half a dozen or more such steppings....

Three basic strategies have been advocated for dealing with the design fault problem for the life-critical system: (1) Testing (Lots of it) (2) Design Diversity (i.e. software fault tolerance: N-version programming, recovery blocks, etc.), and (3) Fault Avoidance (i.e. formal specification/verification, automatic program synthesis, reusable modules). The problem with life testing is that in order to measure ultrareliability one must test for exorbitant amounts of time. For example, to measure a 10^{-9} probability of failure for a 1 hour mission one must test for more than 10^9 hours (114,000 years).

The basic idea of design diversity is to use separate design and implementation teams to produce multiple versions from the same specification. At runtime, non-exact threshold voters are used to mask the effect of a design error in one of the versions. The hope is that the design flaws will manifest errors independently or nearly so. By assuming independence, one can obtain ultrareliable-level estimates of reliability, even with failure rates for the individual versions on the order of 10^{-4}/hour. Unfortunately, the independence assumption has been rejected at the 99% confidence level in several experiments for low reliability software [68, 69].

Furthermore, the independence assumption cannot be validated for high reliability software because of the exorbitant test times required. If one cannot assume independence then one must measure correlations. This is infeasible as well; it requires as much testing time as life-testing the system, because the correlations must be in the ultrareliable region in order for the system to be ultrareliable. Therefore, it is not possible, within feasible amounts of testing time, to establish that design diversity achieves ultrareliability. Consequently, design diversity can create an "illusion" of ultrareliability without actually providing it. For a more detailed discussion, see [14, 13].

1.2 What is Formal Methods

Engineering relies heavily on mathematical models and calculation to make judgments about designs. This is in stark contrast to the way in which software systems are typically designed—with *ad hoc* technique and after-implementation testing. Formal methods bring to software and hardware design the same advantages that other engineering endeavors have exploited: mathematical analysis based on models. Formal methods are used to specify and model the behavior of a system and to formally verify that the system design and implementation satisfy functional and safety properties. Formal methods refers to the use of techniques from logic and discrete mathematics in the specification, design, and construction of computer systems (both hardware and software)[1] and relies on a discipline that requires the explicit enumeration of all assumptions and reasoning steps. Each reasoning step must be an instance of a relatively small number of allowed rules of inference. In essence, system verification is reduced to a calculation that can be checked by a machine. In principle, these techniques can produce error-free design; however, this requires a complete verification from the requirements down to the implementation, which is rarely done in practice.

Thus, formal methods is the applied mathematics of computer systems engineering. It serves a similar role in computer design as Computational Fluid Dynamics (CFD) plays in aeronautical design, providing a means of calculating and hence predicting what the behavior of a digital system will be prior to its implementation.

The tremendous potential of formal methods has been recognized by theoreticians for a long time, but the formal techniques have remained the province of a few academicians, with only a few exceptions such as the Transputer [2] and the IBM CICS project [58]. NASA Langley's program has helped to advance the capabilities of formal methods to the point where commercial exploitation is near.

There are many different types of formal methods with various degrees of rigor. The following is a useful (first-order) taxonomy of the degrees of rigor in formal methods:

Level-1: Formal specification of all or part of the system.

Level-2: Formal specification at two or more levels of abstraction and paper and pencil proofs that the detailed specification implies the more abstract specification.

[1] "Formal" means that the methods of reasoning are valid by virtue of their form and independent of their content.

Level-3: Formal proofs checked by a mechanical theorem prover.

Level 1 represents the use of mathematical logic, or a specification language that has a formal semantics, to specify the system. This can be done at several levels of abstraction. For example, one level might enumerate the required abstract properties of the system, while another level describes an implementation that is algorithmic in style.

Level 2 formal methods goes beyond Level 1 by developing pencil-and-paper proofs that the concrete levels logically imply the abstract, property-oriented levels. *Level 3* is the most rigorous application of formal methods. Here one uses a semi-automatic theorem prover to make sure that all of the proofs are valid. The Level 3 process of *convincing* a mechanical prover is really a process of developing an argument for an ultimate skeptic who must be shown every detail.

It is important to realize that formal methods is not an all-or-nothing approach. The application of formal methods to the most critical portions of a system is a pragmatic and useful strategy. Although a complete formal verification of a large complex system is impractical at this time, a great increase in confidence in the system can be obtained by the use of formal methods at key locations in the system. For more information on the basic principles of formal methods, see [15].

2 Goals of Our Program, Strategy, and Research Team

The major goals of the NASA Langley research program are to make formal methods practical for use on high integrity systems developed in the United States, to orchestrate the transfer of this technology to industry through use of carefully designed demonstration projects, and to exploit this technology to help achieve NASA's ambitious goals in aeronautics. Our intention is to concentrate our research efforts on the technically challenging areas of digital flight-control systems design that are currently beyond the state-of-the-art, while initiating demonstration projects in problem domains where current formal methods are adequate. The challenge of the demonstration projects should not be underestimated. That which is feasible for experts that have developed the tools and methods is often difficult for practitioners in the aerospace industry. There is often a long "learning curve" associated with the tools, the tools are not production-quality, and the tools have few or no examples for specific problem domains. Therefore, we are setting up cooperative efforts between industry and the developers of the formal methods to facilitate the technology transfer process.

This strategy leverages the huge investment of ARPA and the National Security Agency in development of tools and concentrates on the problems specific to the aerospace problem domain. NASA Langley has not sponsored the development of any general-purpose theorem provers. However, the technology transfer projects have lead to significant improvements in the Prototype Verification System (PVS) theorem prover[97] that SRI International (SRI) is developing. Several domain-specific tools have been sponsored: (1) Tablewise, (2) VHDL-analysis tool, and (3) DRS. These tools are discussed in later sections.

It is also important to realize that formal methods include a large class of mathematical techniques and tools. Methods appropriate for one problem domain may be totally inappropriate for other problem domains. The following are some of the specific domains in which our program has concentrated: (1) architectural-level fault tolerance, (2) clock-synchronization, (3) interactive consistency, (4) design of hardware devices such as microprocessors, memory management units, DMA controllers, (5) warning systems for closely spaced parallel landings, (6) design and verification of application-specific integrated circuits (ASICS), (7) Space Shuttle software, (8) navigation software, (9) decision tables, (10) railroad signaling systems, and (11) flight guidance systems.

We are also interested in applying formal methods to many different portions of the life-cycle, such as (1) requirements analysis, (2) high-level design, (3) detailed design, and (4) implementation.

Often, there is a sizable effort associated with the development of the background mathematical theories needed for a particular problem domain. Although such theories are reusable and in the long run can become "cost-effective", the initial costs can be a deterrent for industry. Therefore, one of the goals of the NASA Langley program is to build a large body of background theories needed for aerospace applications.

We also have been involved with standards activities in order to move formal methods technology into digital avionics standards.

2.1 Technology Transfer

The key to successful technology transfer is building a cooperative partnership with an industrial research or development team. We have recognized that in order for such partnerships to work, it is essential that NASA Langley become directly involved in specific problem domains of the aerospace industry. Equally important is the need for industry to make investments to work with NASA on joint projects and help devise realistic and practical demonstration projects. The ultimate goal of our technology transfer process is for formal methods to become the "state-of-the-practice" for developing high integrity digital avionics systems in the United States

Our basic approach to technology transfer is as follows. The first step is to find an industry representative who is interested in formal methods, believes that there is a potential benefit of such methods, and is willing to work with us. The next step is to fund our formal methods research team to apply formal methods to an appropriate application. This process allows the industry representative to see what formal methods are and what they have to offer, and it allows us (the formal methods team) to learn the design and implementation details of state-of-the-practice components so we can better tailor our tools and techniques to industry's needs. If the demonstration project reveals a significant potential benefit, the next stage of the technology transfer process is for the industry representative to initiate an internal formal methods program, and begin a true cooperative partnership with us.

Another important part of our technology transfer strategy is working with the Federal Aviation Administration (FAA) to update certification methods with respect to formal methods. If the certification process can be redefined in a manner that awards credit for the use of formal methods, a significant step towards the transfer of this technology to the commercial aircraft industry will have been accomplished.

Langley has also been sponsoring a series of workshops on formal methods. The first workshop, held in August 1990, focused on building cooperation and communication between U.S. formal methods researchers[18]. The second, held in August 1992, focused on education of the U.S. aerospace industry about formal methods[64]. The third workshop was held in May 1995[52], the fourth in September 1997 [54], and the fifth in June 2000 [53].

Another component of our technology transfer strategy, is to use the NASA's Small Business Innovative Research (SBIR) program to assist small businesses to develop commercially viable formal methods tools and techniques. The first contracts under this program began in early 1994.

Finally, to facilitate technology transfer, much information on NASA Langley's formal methods research is available on the Internet via the World Wide Web at the following location:

http://shemesh.larc.nasa.gov/fm/

3 Recent Technology Development and Transfer Projects

3.1 Formal Analysis of AILS

The Airborne Information for Lateral Spacing (AILS) is a research project within the Reduced Spacing Operations (RSO) element of the Terminal Area Productivity (TAP) Program at NASA. The objective of the AILS research is to increase the ability of aircraft to land on closely-spaced parallel runways in Instrument Meteorological Conditions (IMC). The current minimum runway separation for independent landings during IMC is 4300 feet. Using AILS, independent parallel approaces down to 2500 feet separation are expected to be possible. The AILS system is an airborne alerting system that uses Automatic Dependent Surveillance-Broadcast (ADS-B) datalink and differential GPS.

This inhouse project is exploring the use formal methods to analytically demonstate that the AILS alerting algorithm complies with its requirements for all possible parallel landing scenarios. In particular, the following property is being proved:

> For all possible states s1, s2 and all possible trajectories and assuming that one of the airplanes is in its intended course at the time of the prediction, the algorithm modeled by the function *chtrack* will warn i seconds before a collision.

3.2 SPIDER

The Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER) project is a new project jointly sponsored by NASA and the FAA. The purpose of this project is to develop an advanced fault-tolerant computer system, gain understanding of the new RTCA DO-254 guidance document by developing SPIDER in accordance with its provisions, and generate training materials for the FAA. The RTCA DO-254 document entitled "Design Assurance Guidance for Airborne Electronic Hardware" is intended to provide a basis for the certification of complex electronic hardware devices used in future aircraft.

For the case study, a core subsystem of the Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER) has been selected. SPIDER is a new fault-tolerant architecture under development at NASA Langley Research Center. Several factors motivated the choice of a fault-tolerant system for this exercise. Hardware realizations of fault-tolerant protocols are generally compact designs; this allows for comprehensive treatment within the time constraints of a training exercise. Also, the behavior of fault-tolerant devices is inherently complex; such a device is clearly within the scope of DO-254. Furthermore, there is a considerable amount of research literature addressing the formal analysis of fault-tolerant protocols; a fault-tolerant system is a good candidate for a formal methods demonstration. Finally, any device expected to recover from transient failures will necessarily need to deal with a bounded set of permanent failures, as well.

In the SPIDER architecture, the primary basis for fault-tolerance is a communication subsystem called the Reliable Optical Bus (ROBUS). This concept builds on twenty years of fault-tolerant computing research at NASA Langley Research Center[2]

The SPIDER architecture will be formally modelled and analyzed using a hybrid fault model (See section 4.1.7.) Faulty nodes are globally classified based on the locally observable characteristics to other nodes within the system. The system is partitioned into Fault Containment Regions (FCRs) that ensure independence of random physical failures. The failure status of an FCR is then

[2]The main concept was inspired by a fault-tolerant system designed as part of the Fly-by-Light/Power-by-Wire (FBL/PBW) program.

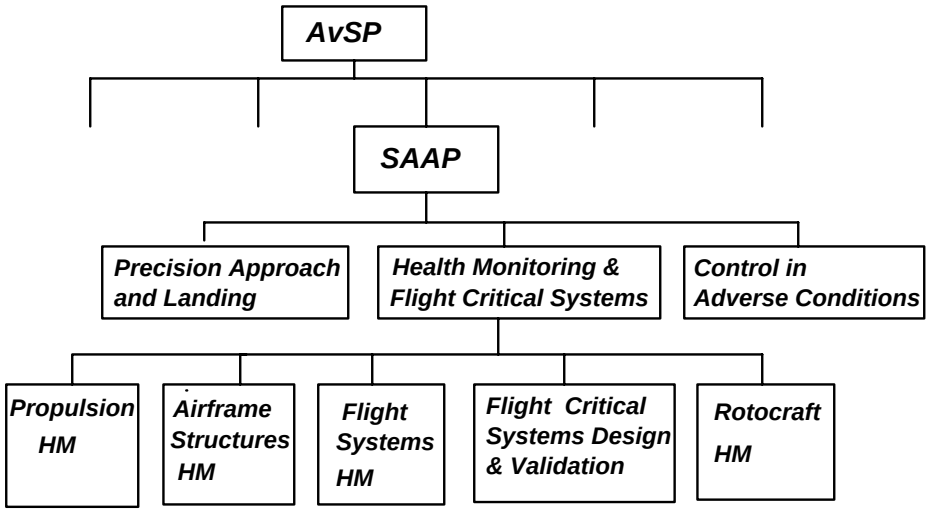
one of four mutually exclusive possibilities:Good, Benign Faulty, Symmetric Faulty, and Asymmetric Faulty. As part of the conceptual design activities, the algorithms for providing the fault-tolerant services are being formally specified and verified using PVS (http://pvs.csl.sri.com/). The focus of the verification activities during the detailed design and implementation stages will be to preserve the integrity of the verification performed at the conceptual design level. We also intend to explore elemental analysis and safety-specific analysis as we proceed but nothing has been done with these techniques yet.

3.3 Aviation Safety Program

A recent competitive procurement (NRA 99-LaRC-4) resulted in 5 awards that will apply formal methods:

- Honeywell Technology Center (Minneapolis): Develop and implement formal verification techniques for verifying the safety of IMA software using the DEOS operating system as the test subject.
- Rockwell Collins Advanced Technology Center: Develop extensions to existing methods and commercial off-the-shelf tools that enable (1) formal requirements modeling and analysis, (2) safety analysis and partitioning, (3) mode confusion detection, and (4) auto-generation of code.
- Honeywell Engines and Systems: Utilize new Time Triggered Architecture (TTA) developed in Europe for the automotive industry and formal verification methods to develop a FTIMA architecture. Targeted application is Full Authority Digital Engine Control (FADEC).
- Barron Associates/BF Goodrich: Develop formal verification methods that can serve as a basis for certifying non-adaptive neural nets for use on military and commercial aircraft.
- University of Virginia/Litton: Identify causes and develop tools to facilitate integration of formal verification methods into the software development lifecycle within an aerospace company.

This work is supported by the Flight Critical Systems Design and Validation component of the Aviation Safety Program (AvSP).



3.4 SpecTRM RL Development and Demonstration

Safeware Engineering Corporation has been awarded a contract to improve their requirements modeling and analysis tool named SpecTRM in the following ways:

- Improvement of the human interface to make SpecTRM easily useable by industry with minimal training.
- Expanded visualizations (both static and dynamic) of SpecTRM models.
- Expand the static analysis capabilities of SpecTRM with model checking and theorem proving
- Application to Space Station Rendezvous and Docking

3.5 Customizable PVS

SRI international (with a subcontract with Honeywell HTC) has been awarded a three-year contract to develop a customizable version of PVS [96, 95, 124].

The contractor intends to accomplish the following:

Year 1 : Provide a more customizable and more open system architecture for PVS; make decision procedures available standalone and demonstrate their use with the Honeywell HOPTs (Hierarchical Operational Procedure Tables) tool for analyzing mode transition tables; undertake a demonstration project, tentatively identified as the Epic ASCB-D synchronization protocol

Year 2 : Develop a methodology and supporting technology for using PVS in conjunction with an industrially accepted engineering notation such as UML; provide a "customizer's" interface.

Year 3 : Large demonstration project with Honeywell: Develop support for other development and assurance processes used in aerospace certification—for example, testing and the MCDC (Modified Condition Decision Coverage) criterion of DO-178B.

Several tutorial introductions to PVS are available [28, 10, 128, 15, 99, 127].

3.6 Timing Analysis by Model Checking

Odyssey Research Associates has been awarded a three-year contract to develop analysis methods for real-time systems analysis. Timing aspects of embedded systems have been notoriously difficult to analyzed and verify. This work will address some of the limitations of Rate Monotonic Analysis (RMA) by applying model checking, a technique with successful industrial applications in hardware design. In addition to schedulability, the new methods will be able to analyze such properties as freedom from deadlock and from certain timing-dependent runtime errors. The goal is both to increase the design space and to reduce the costs of verification. The new methods will be applied to reusable launch vehicle systems. The work will first concentrate on applying the methods to Ada tasking programs executing under a runtime system that meets the Ravenscar Profile.

3.7 PVS Development

NASA Langley's formal methods program has exploited the capabilities of PVS (Practical Verification System) developed by SRI International

http://www.csl.sri.com/sri-csl-fm.html

in no small measure. We have found the tool to be extremely capable, but have also desired improvements. Consequently, NASA Langley has supported the development of abstract datatypes in PVS [100] and the formal sematics for the specification language [123]. In 1996 NASA Langley sponsored the development of a tabular notation for PVS[98]. NASA Langley also supported the development of the PVS validation suite and funded a task to develop an approach for efficient direct execution of PVS specifications that can help users validate their specification through exploration of its behavior on test cases or symbolic execution. With the 2.3 release of PVS, users "run" a specification as if were a program, many times faster than can be achieved using rewriting. Static analysis for live variables allows safe use of destructive updates so that compiled PVS runs at speeds comparable to an imperative program.

Currently NASA Langley is funding SRI to develop an interactive simplifier for PVS[3] and to develop a mechanization for theory interpretations[4]. The developed mechanization will make it possible to show that one collection of theories is correctly interpreted by another collection of theories under a user-specified interpretation. In [97], Rushby describes how NASA's program has shaped the development of PVS.

3.8 Translating UML Into PVS

Odyssey Research Associates has been tasked to develop a UML-based specification and analysis method approporiate for flight guidance systems. The goal is to develop a specification method that uses standard UML notation and develop algorithms to translate these specifications into PVS for analysis. ORA has defined a design style that is strictly hierarchical and local, declarative (i.e. not operational), and object oriented. The defined UML idiom uses restricted class/state diagrams and additional stereotypes for specification. ORA has defined the semantics of the UML subset formally. An executable definition generates formal PVS model of the UML design. The formal model respects the OO structure of the UML.

3.9 Formal Methods Analysis of Mode Confusion

The goal of the new NASA Aviation Safety Program is to reduce the civil aviation fatal accident rate by 80% in ten years and 90% in twenty years. This program is being driven by the accident data with a focus on the most recent history. Pilot error is the most commonly cited cause for fatal accidents (up to 70%), and thus is being given major consideration in this program. The January 30, 1995 issue of Aviation Week lists 184 incidents and accidents involving mode awareness including the Bangalore A320 crash (14 February 1990), the trasbourg A320 crash (20 January 1992), the Habsheim A320 crash (26 June 1988), and the Toulouse A330 crash (30 June 1994) [2]. These incidents and accidents reveal that pilots sometimes become confused about what the

[3]Given $x > y$, $z > 0$, it is surprisingly hard to prove $x/z > y/z$ in PVS. Since this is outside the domain if the decision procedures, must find a lemma in prelude and instantiate correctly. But if we could say "multiply both sides by z", it would be easy. The simplification will replace a subexpression by a simpler one that is equivalent or stronger. Currently, simplification in PVS is automatic: hard to understand and to control

[4]The mechanization shall make it possible to show that one collection of theories is correctly interpreted by another collection of theories under a user-specified interpretation. The mechanization will generate the proof obligations necessary to ensure the appropriate relationship, and provide theorem-proving enhancements (for example, rewriting under congruences, and congruence closure) that enable these obligations to be discharged efficiently. The PVS grammar will been extended so that names can include mappings. The goal is to map uninterpreted types and constants of a source theory into types and constants, respectively, of another theory.

cockpit automation is doing. Consequently, human factors research is an obvious investment area. However, even a cursory look at the incident and ccident data reveals that the mode confusion problem is much deeper than just training deficiencies and a lack of human-oriented design. This is readily acknowledged by human factors experts. For example, Charles E. Billings, writes in Aviation Automation: The Search for a Human-Centered Approach, 1997 (pg 144):

> ... today's flight management systems are "mode rich" and it is often difficult for pilots to keep track of them (see Fig 9.2). The second problem, which is related to the first involves lack of understanding by pilot's of the system's internal architecture and logic, and therefore a lack of understanding of what the machine is doing, and why, and what it is going to do next.

Similarly, Sarter and Woods write in Decomposing Automation (1994):

> What is needed is better understanding of how the machine operates, not just how to operate the machine.

It seems that further progress in human factors will only come through a deeper scrutiny of the internals of the automation. Formal methods can contribute in this arena. The fundamental goal of formal methods is to capture requirements, designs, and implementations in a mathematically-based model that can be analyzed in a rigorous manner. By capturing the internal behavior of a flight deck in a rigorous and detailed formal model, the dark corners of a design can be analyzed.

This project is exploring two complementary strategies based on a formal model:

- Visualization: Create and display a clear, executable formal model of the automation that is easily understood by flight crew and use it to drive the flight deck simulation during training.
- Analysis: Conduct mathematical analysis of the model and search for mode confusion potential.

The first phase of a new project involving NASA Langley and Rockwell Collins in applying formal methods to a realistic business jet flight guidance system has been completed and was reported at DASC'98 [16]. A final report on the phase I work has been published[80].

3.10 Formal Analysis of Avionics Partitioning

The RTCA Special Committee 182 (SC-182) has been established to develop a Minimum Operational Performance Standard (MOPS) for an Avionics Computer Resource (ACR). The ACR will have the capability of performing multiple aircraft functions through use of partitioning. Fundamental to the success of the ACR strategy is a guarantee that the ACR platform will prevent any application from corrupting another. In particular, the ACR must provide:

- space partitioning (no matter what an application does it cannot corrupt the memory of another application), and
- time partitioning (no matter what an application does it cannot prevent another application from obtaining its scheduled allocation of CPU time)

while supporting:

- inter-partition communications, and

- common modular/configurable I/O (the ACR must allow the partitions access to external devices through a suite of bus protocols such as ARINC 429, ARINC 629, etc.)

Our first efforts have been directed towards developing an abstract formal model of space partitioning that can serve as the basis for evaluating the design of an ACR[32]. The PVS formal model is based on mathematical modeling techniques developed by the computer security community. The model has been used on three candidate designs, each an abstraction of features found in real systems.

SRI International was funded by the FAA and NASA to identify the requirements for partitioning in integrated modular avionics (IMA) and to explore how to achieve those requirements with very high assurance. The final report entitled " Partitioning in Avionics Architectures: Requirements, Mechanisms and Assurance" has been published [114].

3.11 Streamlining Software Aspects of Certification

Kelly Hayhurst served as the Technical Program Manager for a FAA initiative called Streamlining Software Aspects of Certification (SSAC) starting in November 1997. The goal of the SSAC program was to identify and eliminate unnecessary costs in software aspects of certification for both airborne and ground-based systems. Unnecessary costs in certification not only waste money, but they also can delay adopting new, safety-enhancing technologies.

Since January 1998, the SSAC team has conducted 3 industry workshops and 1 FAA workshop to identify specific software issues to be addressed by the program. The team also conducted a survey of U.S. companies about the extent and significance of the issues identified through the workshops. A report describing the results and recommendations was published and is available electronically. The FAA response to the survey recommendations and the letter accompanying this response are also available in PDF format.

In October 1999, the FAA discontinued funding for the SSAC team activities. However, there continues to be interest in the SSAC program results and recommendations. A presentation on the current status of the SSAC program was given at the FAA National Software Conference on April 20, 2000. See

`http://shemesh.larc.nasa.gov/ssac/`

to obtain details about these workshops and copies of the workshop reports. A published report [51] about the project is available electronically at

`http://shemesh.larc.nasa.gov/ssac/workshop3.html`

The FAA response to the survey recommendations and the letter accompanying this response are also available in PDF format at this web page.

3.12 ICASE Research

The Formal Methods research program at NASA Langley's Institute For Computer Applications in Science and Engineering (ICASE) is developing and applying techniques and tools for the specification, analysis, and verification of aerospace digital systems. Research efforts focus especially on the investigation of heterogeneous verification technologies in order to enhance the utility of formal methods for specifying, analyzing, and verifying large-scale systems. Topics of interest/under investigation include:

- applying state-exploration methods to the verification of flight-guidance systems and to the analysis of mode confusion
- formalizing the Airborne Information System for Lateral Spacing (AILS)
- developing compositional approaches to Statecharts semantics (including UML Statecharts)
- extending static type checking in programming languages (such as Java)
- designing efficient higher-order unification algorithms for theorem proving
- combining process algebras and temporal logics

The following are recent publications:

> Cesar Munoz and Victor Carreno, Aircraft trajectory modeling and alerting algorithm verification, NASA/CR-2000-210097 ICASE Report No. 2000-16, April 2000, pp. 25.
>
> Gerald Luettgen, Michael von der Beeck, and Rance Cleaveland, A Compositional approach to Statecharts semantics, NASA/CR-2000-210086 ICASE Report No. 2000-12, March 2000, pp. 20.
>
> Rance Cleaveland and Gerald Luettgen, Model checking is refinement - Relating Buechi testing and linear-time temporal logic, NASA/CR-2000-210090 ICASE Report No. 2000-14, March 2000, pp. 25.

3.13 NASA Small Business Innovative Research Program

In an effort to encourage small businesses to develop good formal methods tools, we have participated in the NASA Small Business Innovative Research (SBIR) program since 1993. The SBIR program provides the opportunity for small businesses to submit proposals in various research areas each year. In Phase 1, companies whose proposals are chosen in a NASA-wide selection process are awarded a 6-month contract to conduct initial research and feasiblity studies. At the end of this period, the companies submit a Phase 2 proposal for up to two years of additional research and development. Companies whose proposals are chosen in this NASA-wise selection process are awarded a two year contract to complete the work proposed. More information on NASA's SBIR program is available on the World-Wide Web at

> `http://www.sbir.nasa.gov/.`

Since our participation in the SBIR program began, we have advocated eight formal methods proposals for Phase I funding; five of these have been chosen by NASA to receive funding. Two of the three non-funded proposals were for work substantially similar to work that was eventually funded, so we have been very successful in advocating those proposals that we like.

We have recommended four of five proposals for Phase II funding. For one, the Phase II selection process has not be completed. The other three have all been chosen by NASA to receive funding. The following list shows the year of selection, proposal title, and company for each of these three:

1. 1998, Multi-Formal Hardware Verification System, Levetate Design Systems, Inc.
   (http://www.levetate.com/)
2. 1997, Verified VHDL Synthesizable Cores, Derivation Systems Inc.
   (http://www.derivation.com/)

3. 1994, Analysis Tools for VHSIC Hardware Description Language, Odyssey Research Associates, Inc. (http://www.oracorp.com/)

4. 1994, Digital Design Derivation System for Hardware Synthesis, Derivation Systems, Inc. (http://www.derivation.com/)

4 Past Efforts

This section describes previous work in three categories: technology transfer, fault-tolerant systems, other fundamental research.

4.1 Technology Transfer

4.1.1 AAMP5/AAMP-FV Project

In 1993, NASA Langley initiated a joint project involving Collins Commercial Avionics and SRI International. The goal was to investigate the application of formal techniques to a commercial microprocessor design, the Collins AAMP5 microprocessor. The AAMP5 is the latest member of the CAPS/AAMP family of microprocessors and is object code compatible with the AAMP2 processor [3]. The CAPS/AAMP family of microprocessors has been widely used by the commercial and military aerospace industries. Some examples of use of earlier members of the family include: (1) Boeing 747-400 IDS, (2) Boeing 737-300 EFIS, and (3) Boeing 757,767 AFDS

The first phase of the project consisted of the formal specification of the AAMP5 instruction set and microarchitecture using SRI's PVS. While formally specifying the microprocessor, two design errors were discovered in the microcode. These errors were uncovered as a result of questions raised by the formal methods researchers at Collins and SRI while seeking to formally specify the behavior of the microprocessor[79, 132]. The Collins formal methods team believes that this effort has prevented two significant errors from going into the first fabrication of the microprocessor.

The second phase of the project consisted of formally verifying the microcode of a representative subset of the AAMP5 instructions. Collins seeded two errors in the microcode provided to SRI in an attempt to assess the effectiveness of formal verification. Both of these errors (and suggested corrections) were discovered while proving the microcode correct[79]. It is noteworthy that both the level 2 and level 3 applications of formal methods were successful in finding bugs. Based on the success of the AAMP5 project, a new effort was initiated with Rockwell-Collins to apply formal methods in the design level verification of a microprocessor, currently designated as AAMP-FV. This project is nearing completion.

This work has has significant impact on the development of a hardware verification capability within PVS [30]. This project has also had major impact on Rockwell Collins. There are now four engineers at Collins that are skilled in formal methods. In the fall of 1996 Rockwell Collins hired a formal methods expert whose full-time job is to integrate the use of formal methods into their product lines.

Rockwell Collins used the formal specifications of the AAMP5/AAMP-FV micro-architecture as a means to design and perform prototype testing on their JEM1 Java chip. Collins was the first supplier to SUN Microsystems to deliver a microprocessor that directly executes the JAVA instruction set.

4.1.2 Tablewise Project

Under NASA funding, Odyssey Research Associates worked with Honeywell Air Transport Systems Division (Phoenix) beginning in 1993 to study the incorporation of formal methods into the company's software development processes. As part of this work, ORA developed a prototype tool, called TableWise, to analyze the characteristics of certain types of operational procedure tables. An operational procedure table is a tablar format for defining the rules that choose a particular action to perform based on the values of certain parameters; they are equivalent in expressive power to finite state machines.

ORA concentrated on a subset of operational procedure tables, called decision tables. TableWise uses a generalization of Binary Decision Diagrams to determine if a decision table is exclusive (for every combination of parameter values, at most one action can be chosen) and exhaustive (for every combination of parameter values, at least one action can be chosen). The tool is also capable of automatically generating documentation and Ada code from a table[56]. ORA also investigated methods for generating software test cases directly from tables[55], and for reverse engineering tabular specifications from existing avionics code.

Boeing Defense and Space Systems has funded ORA to develop an extension to Tablewise: to enable it to reason about arithmetic expressions that are commonplace in navigation system decision logic at Boeing. ORA is also building an interface between Tablewise and Boeing's GSDS (an internal CAD tool) system. It will convert GSDS STM (State Transition Matrix) tables to tablewise tables and will allow GSDS designers to check Boeing STM tables for completeness and consistency.

4.1.3 Space Shuttle Change Requests

A team spread across three NASA centers (LaRC, JSC, and JPL), together with support from Loral Space Information Systems [5] , SRI International, and VíGYAN Inc., was formed to study the application and technology transfer of formal methods to NASA space programs from 1992 until 1996. The NASA Formal Methods Demonstration Project for Space Applications focused on the use of formal methods for requirements analysis because the team believed that formal methods are more practically applied to requirements analysis than to late-lifecycle development phases [61]. This proved to be a valuable decision.

In 1993 a formal specification of a very mature piece of the Space Shuttle flight control requirements called Jet Select was developed. Few proofs were produced for the first specification, but 46 issues were identified and several minor errors were found in the requirements. A second specification was produced for an abstract (i.e., high level) representation of the Jet Select requirements. This abstraction, along with the 24 proofs of key properties, was accomplished in under 2 work months, and although it only uncovered 6 issues, several of these issues were significant [26].

NASA Langley's primary role in 1994–95 included support for three Space Shuttle software change requests (CRs). One CR concerned the integration of new Global Positioning System (GPS) functions, while a second CR concerned a new function to control contingency aborts known as Three Engine Out (3E/O), and the third CR concerned improved crew displays when entering the "heading alignment cylinder" (HAC) during landing.

Due to the size and complexity of the GPS CR, the trial formal methods task focused on just a few key areas. Formal specifications were developed for the new Shuttle navigation "principal functions" known as GPS Receiver State Processing and GPS Reference State Processing, using the

[5] now Lockheed Martin Space Information Systems

language of SRI's Prototype Verification System (PVS). The specifications were revised twice to reflect major requirements changes during the study period. As a result of the three formalization efforts, a total of 86 issues or minor discrepancies were discovered in the CR. These items were submitted as official issues during three requirements inspections, leading to a favorable reaction from the Shuttle requirements community [37, 31].

The Three Engine Out (3E/O) Task is executed each cycle during powered flight until either a contingency abort maneuver is required or progress along the powered flight trajectory is sufficient to preclude a contingency abort even if three main engines fail. We developed and analyzed a formal model of the series of sequential maneuvers that comprise the 3E/O algorithm. A total of 19 potential issues were found, including undocumented assumptions, logical errors, and inconsistent and imprecise terminology. The formal specifications were analyzed using the Murϕ system [38], which performs analysis of finite state systems using model checking. The results of this study are reported in [27].

Our role in support of the HAC CR was much smaller than for GPS or 3E/O. Key portions of the the control logic for managing the HAC crew displays are expressed in the CR in a tabular form. Two HAC tables were analyzed for consistent and complete enabling conditions using a new PVS feature for analyzing tabular specifications. One table was found to have overlapping conditions, meaning that several rows were not disjoint. A corrected version was produced and both tables were then proved to satisfy the well-formedness criteria using PVS [105].

In addition to the Shuttle activities, LaRC contributed to two NASA guidebooks produced by the inter-center team. The first volume of the guidebook, published in 1995, is intended for managers of NASA projects who will be using formal methods in requirements analysis activities [92]. The second volume was published in 1997 and was aimed at practitioners [93]. LaRC's efforts in 1996 were directed primarily at this second guidebook volume, with members of the LaRC team having been the lead contributors.

4.1.4 Union Switch and Signal

As part of a joint research agreement, NASA Langley formal methods researchers collaborated with engineers at Union Switch and Signal (US&S) to use formal methods in the design of railway switching and control applications. Railway switching control systems, like digital flight control systems, are safety critical systems. US&S is the leading U.S. supplier of railway switching control systems.

The initial project, started in 1993, was a cooperative effort between NASA, US&S, and Odyssey Research Associates (ORA). The result of this first project was a formal mathematical model of a railway switching network, defined in two levels. The top level of the model provides the mechanisms for defining the basic concepts: track, switches, trains and their positions and control liners of a train (i.e. how far down the track it has clearance to travel.) The second level is a formalization of the standard scheme used in railroad control, the block model control system. A level 2 proof that the fixed block control system is "safe" with respect to the top level model was also completed [57]

Work between US&S, ORA, and NASA concentrated on safety issues within the context of a CAD system begin developed by US&S for use by railway control engineers. An area of particular concern has been the correctness of internal compilation processes which translate graphical representations of control diagrams into code that will be executed on US&S's proprietary V_FRAME^{++} architecture.

4.1.5 Honeywell Navigation Specification

A cooperative research effort was initiated in 1993 with Honeywell Air Transport Systems Division (Phoenix) to study the incorporation of formal methods into the company's software development processes. In the initial project in this effort, NASA Langley funded ORA to identify a component of the Boeing 777 system to which formal specification techniques could be applied, and to develop the formal specifications for that component. ORA, in collaboration with personnel from Langley and Honeywell, chose the navigation subsystem as a suitable application.

Using documents supplied to them by Honeywell, ORA developed a specification that addressed the following aspects of navigation: (1) basic mathematical concepts such as functions over the reals, and physical units such as distance, velocity, and acceleration, (2) definition of objects such as aircraft, radios, sensors, navigation aids, and the navigation database, (3) definition of algorithms such as complementary filter processing, navigation aid selection, navigation mode selection, and position determination, and (4) relating the mathematical model to Ada by partitioning the system in Ada package specifications, and annotating individual Ada functions and procedures with formal specifications. The specification was done using ORA's Penelope tool.

4.1.6 CSDL Scoreboard Hardware

A joint project between ORA and Charles Stark Draper Laboratory (CSDL) was completed in 1993. NASA Langley and the Army had funded CSDL to build advanced, fault-tolerant computer systems for over two decades. During this time, CSDL became interested in the use of formal methods to increase confidence in their designs. ORA was given the task of formally specifying and verifying a key circuit (called the scoreboard) of the Fault-Tolerant Parallel Processor (FTPP) [50] in Clio [131]. The formal verification uncovered previously unknown design errors. When the scoreboard chip was fabricated, it worked without any error manifestation. It was the first time that CSDL produced a chip that worked "perfectly" on a first fabrication. CSDL credits VHDL-development tools and formal methods for the success.

4.1.7 Allied Signal's Hybrid Fault Algorithms

Thambidurai and Park (Allied-Signal) introduced a hybrid fault model (1988) that classified faults into three categories: asymmetric, symmetric and crash. They further suggested the need for and developed an algorithm that had capabilities beyond that of the earlier Byzantine generals algorithms. In particular, their algorithm can mask the effects of a less severe class of faults, in a more effective way[6]. A formal analysis by SRI discovered flaws in Allied-Signal's algorithm Z and together with Allied Signal, they developed an improved algorithm [76, 75, 77].

The newly developed hybrid-fault theory was then applied to the analysis of the Charles Stark Draper Labs "Fault-Tolerant Processor" (FTP). A unique feature of this architecture is its use of "interstages" to relay messages between processors. These are significantly smaller than a processor and lead to an asymmetric architecture that is far more efficient than the traditional Byzantine agreement architectures[7]. The SRI work not only formalized the existing informal analysis but extended it to cover a wider range of faulty behavior[78]. Also, SRI generalized their clock synchronization work to encompass the hybrid fault model [110].

[6] This was done during the development of the [Multicomputer Architecture for Fault Tolerance (MAFT) system [133].

[7] This combination of algorithm, architecture, and fault model represents the best known compromise between economy and fault tolerance. Other combinations either tolerate less faults, or less severe kinds of faults, for a given level of redundancy, or require more hardware to tolerate the same number and kinds of faults.

Next, SRI investigated authenticated Byzantine Agreement while extending fault model to include link failures as well as hybrid faults in the processors [46]. The analysis was performed using both the PVS theorem proving system and model checking (Stanford Murϕ). Tradeoffs between different algorithms were explored via symbolic fault-injection with the Murϕ Tool. There is currently much interest in combining model checking and general purpose theorem proving. Some effort in this direction has been sponsored by the NASA program [104, 29].

Other work by SRI applied these ideas to reconfigurable, fault-tolerant systems [112].

4.2 Fault-tolerant Systems

The goal of this focus area was to create a formalized theory of fault tolerance including redundancy management, clock synchronization, Byzantine agreement, voting, etc. Much of the theory developed here is applicable to future fault-tolerant systems designs. A detailed design of a fault-tolerant reliable computing base, the Reliable Computing Platform (RCP), has been developed and proven correct. It is hoped that the RCP will serve as a demonstration of the formal methods process and provide a foundation that can be expanded and used for future aerospace applications. It is one of the largest formal verifications ever performed.

The RCP architecture was designed in accordance with a system-design philosophy called "Design For Validation" [63, 62].

A major objective of this philosophy is to minimize the amount of experimental testing required and maximize the ability to reason mathematically about correctness of the design. Although testing cannot be eliminated from the design/validation process, *the primary basis of belief in the dependability of the system must come from analysis rather than from testing.*

4.2.1 The Reliable Computing Platform

The Reliable Computing Platform dispatches control-law application tasks and executes them on redundant processors. The intended applications are safety critical with reliability requirements on the order of $1 - 10^{-9}$. The reliable computing platform performs the necessary fault-tolerant functions and provides an interface to the network of sensors and actuators.

The RCP operating system provides the applications software developer with a reliable mechanism for dispatching periodic tasks on a fault-tolerant computing base that *appears* to him as a single ultrareliable processor. A multi-level hierarchical specification of the RCP is shown in figure 1.

The top level of the hierarchy describes the operating system as a function that sequentially invokes application tasks. This view of the operating system will be referred to as the *uniprocessor specification (US)*, which is formalized as a state transition system and forms the basis of the specification for the RCP. Fault tolerance is achieved by voting results computed by the replicated processors operating on the same inputs. Interactive consistency checks on sensor inputs and voting of actuator outputs require synchronization of the replicated processors. The second level in the hierarchy (RS) describes the operating system as a synchronous system where each replicated processor executes the same application tasks. The existence of a global time base, an interactive consistency mechanism and a reliable voting mechanism are assumed at this level.

Level 3 of the hierarchy (DS) breaks a frame into four sequential phases. This allows a more explicit modeling of interprocessor communication and the time phasing of computation, communication, and voting. At the fourth level (DA), the assumptions of the synchronous model must be discharged. Rushby and von Henke [115] report on the formal verification of Lamport and Melliar-Smith's [71] interactive-convergence clock synchronization algorithm. This algorithm can

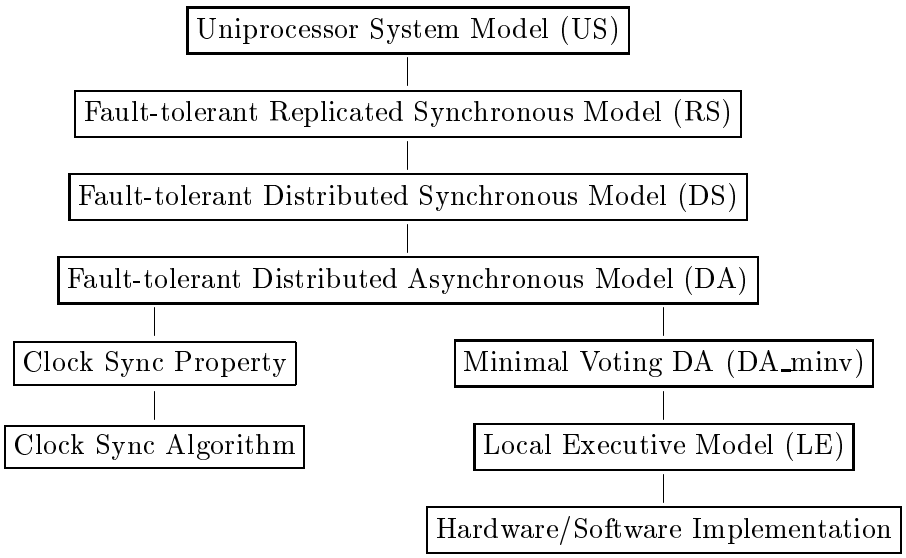


Figure 1: Hierarchical Specification of the Reliable Computing Platform.

serve as a foundation for the implementation of the replicated system by bounding the amount of asynchrony in the system so that it can duplicate the functionality of the DS model. Dedicated hardware implementations of the clock synchronization function are a long-term goal.

In the LE model, a more detailed specification of the activities on a local processor are presented. In particular, three areas of activity are elaborated in detail: (1) task dispatching and execution, (2) minimal voting, and (3) interprocessor communication via mailboxes. An intermediate model, DA_minv, that simplified the construction of the LE model was used. Some of the refinements occur in the DA_minv model and some in the LE model. For example, the concept of minimal voting is addressed in considerable detail in the DA_minv model. Of primary importance in the LE specification is the use of a memory management unit by the local executive in order to prevent the overwriting of incorrect memory locations while recovering from the effects of a transient fault.

The top two levels of the RCP were originally formally specified in standard mathematical notation and connected via mathematical (i.e. level 2 formal methods) proof [36, 35, 33]. Under the assumption that a majority of processors is working in each frame, the proof establishes that the replicated system computes the same results as a single processor system not subject to failures. Sufficient conditions were developed that guarantee that the replicated system recovers from transient faults within a bounded amount of time. SRI subsequently generalized the models and constructed a mechanical proof in EHDM [107]. Next, the local team developed the third and fourth level models. The top two levels and the two new models (i.e. DS and DA) were then specified in EHDM and all of the proofs were done mechanically using the EHDM 5.2 prover [11, 34].

Both the DA_minv model and the LE model were specified formally and have been verified using the EHDM verification system[12]. All RCP specifications and proofs are available electronically via the Internet using anonymous FTP or World Wide Web (WWW) access. Anonymous FTP access is available through the host `deduction.larc.nasa.gov` using the path `pub/fm/larc/RCP-specs`. WWW access to the FTP directory is provided through the NASA Langley Formal Methods Program home page: `http://atb-www.larc.nasa.gov/fm.html`

Two recent publications by SRI provide some new theoretical insights [112, 113].

4.2.2 Clock Synchronization

The redundancy management strategies of virtually all fault-tolerant systems depend on some form of voting, which in turn depends on synchronization. Although in many systems the clock synchronization function has not been decoupled from the applications (e.g. the redundant versions of the applications synchronize by messages), research and experience have led us to believe that solving the synchronization problem independently from the applications design can provide significant simplification of the system [70, 45]. The operating system is built on top of this clock-synchronization foundation and thus the correctness of this foundation is essential. The clock synchronization algorithm and its implementation are prime candidates for formal methods. The verification strategy shown in figure 2 is being explored.

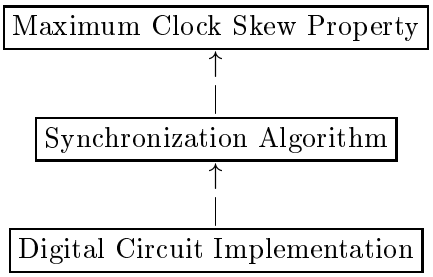


Figure 2: Hierarchical Verification of Clock Synchronization

The top-level in the hierarchy is an abstract property of the form:

$$\forall \text{ non-faulty } p, q : |C_p(t) - C_q(t)| < \delta$$

where δ is the maximum clock skew guaranteed by the algorithm as long as a sufficient number of clocks (and the processors they are attached to) are working. The function $C_p(t)$ gives the value of clock p at real time t. The middle level in the hierarchy is a mathematical definition of the synchronization algorithm. The bottom level is a detailed digital design of a circuit that implements the algorithm. The bottom level is sufficiently detailed to make translation into silicon straight forward.

The verification process involves two important steps: (1) verification that the algorithm satisfies the maximum skew property and (2) verification that the digital circuitry correctly implements the algorithm. The first step was completed by SRI International. The first such proof was accomplished during the design and verification of SIFT [71]. The proof was done by hand in the style of journal proofs. More recently this proof step was mechanically verified using the EHDM theorem prover[115, 116]. In addition, SRI mechanically verified Schneider's clock synchronization paradigm [117] using EHDM[125, 126]. A further generalization was found at NASA Langley [81][8]. The design of a digital circuit to distribute clock values in support of fault-tolerant synchronization was completed by SRI and was partially verified.[9] CLI reproduced the SRI verification of the interactive convergence algorithm using the Boyer-Moore theorem prover [139].

NASA Langley researchers designed and implemented a fault-tolerant clock synchronization circuit capable of recovery from transient faults [83, 82, 81]. The top-level specification for the design is the EHDM verification of Schneider's paradigm. The circuit was implemented with programmable logic devices (PLDs) and FOXI fiber optic communications chips [88].

[8] The bounded delay assumption was shown to follow from the other assumptions of the theory.

[9] Unlike the NASA circuit, the SRI intent is that the convergence algorithm be implemented in software.

Using a combination of formal techniques, a verified clock synchronization circuit design has also been developed[89]. The principal design tool was the Digital Design Derivation system (DDD) developed by Indiana University[7]. Some design optimizations that were not possible within DDD were verified using PVS.

4.2.3 Byzantine Agreement Algorithms

Fault-tolerant systems, although internally redundant, must deal with single-source information from the external world. For example, a flight control system is built around the notion of feedback from physical sensors such as accelerometers, position sensors, and pressure sensors. Although these can be replicated (and they usually are), the replicates do not produce identical results. To use bit-by-bit majority voting, all of the computational replicates must operate on identical input data. Thus, the sensor values (the complete redundant suite) must be distributed to each processor in a manner which guarantees that all working processors receive exactly the same value even in the presence of some faulty processors. This is the classic Byzantine Generals problem [72]; algorithms to solve the problem are called Byzantine agreement algorithms. CLI investigated the formal verification and implementation of such algorithms. They formally verified the original Marshall, Shostak, and Lamport version of this algorithm using the Boyer Moore theorem prover [4]. They also implemented this algorithm down to the register-transfer level and demonstrated that it implements the mathematical algorithm [5], and then subsequently verified the design down to a hardware description language HDL developed at CLI [91]. A more efficient mechanical proof of the oral messages algorithm was also developed by SRI[108].

ORA also investigated the formal verification of Byzantine Generals algorithms. They focused on the practical implementation of a Byzantine-resilient communications mechanism between Mini-Cayuga micro-processors [130, 6]. The Mini-Cayuga is a small but formally verified microprocessor developed by ORA. It was a research prototype and was not fabricated.

4.3 Other Fundamental Research

4.3.1 Efficient Validation of Superscalar Microprocessors

SRI has developed a new approach to decompose and incrementally build the proof of correctness of pipelined microprocessors. The approach centers around the construction of an abstraction function using *completion functions*, one per unfinished instruction in the pipeline. In addition to avoiding the term size and case explosion problem that limits the pure flushing approach, the new method helps localize errors and handles stages with iterative loops. A final report has been written and is under review to be published as a NASA CR [129].

4.3.2 Specification of Floating-point Arithmetic

Significant portions of the ANSI/IEEE-854 [60] standard have been defined using the PVS [84] and HOL [19] systems. IEEE-854 is a standard for radix-independent floating-point arithmetic. The main motivating factors for the formalization of the standard are 1) The creation of a formal specification against which an implementation (such as the AAMP5 [79]) could be verified; 2) The highly publicized floating-point divide flaw in the Intel Pentium (R) processor [118].

The formalization of the standard has brought to light the interesting and challenging issues of translating a natural language document into a logic based language in a precise, unambiguous, and accurate manner. In addition, the formalization of the standard in two different systems has given the opportunity to compare the verification systems and specification styles [23].

A parameterized formal theory of subtractive floating point division algorithms has been developed in the PVS specification language. This parameterized theory defines a general algorithm that covers a broad class of algorithms and can serve as a tool for design tradeoffs. This generalized theory has been formally proved to satisfy the IEEE-854 standard for floating point arithmetic. The proof covers the entire class of algorithms. Thus, after an optimal algorithm has been selected for a particular target technology, and the parameters are shown to be type-correct, one knows that your selected algorithm is formally consistent with the IEEE standard. This work was presented at the Formal Methods in Computer-Aided Design Conference (FMCAD) on Nov. 6-8, 1996 [87].

4.3.3 Hardware Verification Using Coinduction

Hardware development today is fundamentally dependent upon the use of simulation models to gain confidence in prototype designs. Billions of test cases are run on the prototype models in order to cover the increasingly large input space. If the verification process fails to detect errors, the consequence can be expensive. The Pentium floating-point division bug cost Intel $475 million.

In this project a different approach was pursued: construct a formal model of design and prove mathematically that design computes intended function for complete input space. Unfortunately there are have been many technical problems associated with this approach in the past. Often the initial value of state had to be retained in expression, even when they are no longer relevant. The formal models included the number of clock ticks since start of computation, which made it difficult to verify revisions to a design, particular for aggressive optimizations. Also the formal models restricted the design space available and verification was still expensive.

This inhouse project sought to overcome these limitations through use of a better formal model: stream equations. The designs, which are represented as system of equations, can be refined using algebraic transformations. The state component represents the current state and not the initial state and there is no explicit clock parameter. There is also a significant potential for a much more efficient verification process. Stream equations lead to a corecursive definition of hardware behavior and hence allows verification by coinduction, a very efficient approach for verifying design optimizations.

A general mechanized support for reasoning about streams (within theorem proving system PVS was developed and demonstrated on two significant case studies: (1) Fault-Tolerant Clock Synchronization Circuit [89, 86] and, (2) Floating Point Division[85] .

4.3.4 PVS Libraries

There is a sizable effort associated with the development of the background mathematical theories needed for any particular problem domain. These libraries provide fundamental definitions that are usually taken for granted by a domain expert. For example, fault-tolerant systems rely on voting which requires a theory about the majority function. The library must provide a definition of majority and proven lemmas for all of the commonly used properties. These libraries are reusable and in the long run are cost-effective; but, the initial costs are a deterrent for industry. Therefore, members of the local staff are building a large body of background theories needed for aerospace applications. The following libraries have been constructed and are available over the internet: (1) bitvectors library for hardware verification, (2) IEEE floating point standard, (3) finite sets, (4) div and mod over integers, (5) min/max, majority and sorting over arrays and finite sequences, (6) summations, (7) real analysis, (8) bags, and (9) elementary graph theory.

4.3.5 Formal Modeling of Dynamic Systems

Transition Assertions[20, 21] is an experimental modeling method to represent or specify real-time systems. The model is intended to be used where timing is a critical element. Timing constrains can be represented directly in the model and verified using mathematical logic. Variables and conditions in a system are functions from time to value.

A system is specified by using transition templates which describe a cause and effect relation of the system. Each transition template consists of a durational lower and upper bound, enabling predicate, and execution predicate. Ten generic transition templates have been created featuring single transition, multiple transition, transition with preemption, transition with inertia, and a combination of these.

4.3.6 Verification of Existing Ada Applications Software

Odyssey Research Associates completed two tasks applying their Ada verification tools to aerospace applications. The first task was to verify some utility routines obtained from the NASA Goddard Space Flight Center and the NASA Lewis Research Center using their Ada Verification Tool named Penelope [47]. This task was accomplished in two steps: (1) formal specification of the routines and (2) formal verification of the routines. Both steps uncovered errors [39]. The second task was to formally specify the mode-control panel logic of a Boeing 737 experimental aircraft system using Larch (the specification language used by Penelope) [48].

4.3.7 Boeing Hardware Devices

The Boeing Company was contracted by NASA Langley to develop advanced validation and verification techniques for fly-by-wire systems. As part of the project, Boeing explored the use of formal methods. The goal of this work was two-fold: (1) technology transfer of formal methods to Boeing, and (2) assessment of formal methods technology maturity.

The first phase of this project focused on the formal verification of "real" hardware devices using the HOL hardware verification methodology. With the assistance of a subcontract with U. C. Davis, Boeing partially verified a set of hardware devices, including a microprocessor[137], a floating-point coprocessor similar to the Intel 8087 but smaller[102, 101], a direct memory access (DMA) controller similar to the Intel 8237A but smaller[67], and a set of memory-management units[122, 119]. U. C. Davis also developed the generic-interpreter theory to aid in the formal specification and verification of hardware devices[138, 136, 135], and a horizontal-integration theory for composing verified devices into a system[121, 120, 103, 66]. After demonstrating the feasibility of verifying standard hardware devices, Boeing applied the methodology to a proprietary hardware device called the Processor Interface Unit (PIU) that is being developed for aeronautics and space applications[42].

Boeing and U.C. Davis also performed an assessment of the U.K. Royal Signals and Radar Establishment's (RSRE) VIPER chip [74]. This was part of a now-completed 3 year Memorandum of Understanding (MOU) with RSRE. CLI and Langley researchers also performed assessments of the VIPER project[9, 22, 17]. Application of formal methods to the suite of Intel-like devices and the PIU demonstrated that formal methods can be practically applied to the digital hardware devices being developed by Boeing today and provided insight on how to make the process more cost effective.

4.3.8 Asynchronous Communication

CLI developed a formal model of asynchronous communication and demonstrated its utility by formally verifying a widely used protocol for asynchronous communication called the bi-phase mark protocol, also known as "Bi-Φ-M," "FM" or "single density" [90]. It is one of several protocols implemented by microcontrollers such as the Intel 82530 and is used in the Intel 82C501AD Ethernet Serial Interface.

4.3.9 Digital Design Derivation

Funded in part by a NASA Langley Graduate Student Research Program fellowship, Bhaskar Bose developed the Digital Design Derivation system (DDD) and used it to design a verified microprocessor. DDD implements a formal design algebra that allows a designer to transform a formal specification into a correct implementation[7]. Bose formally derived the DDD-FM9001[8] microprocessor from Hunt's top-level specification of the FM9001 microprocessor[59].

4.3.10 Civil Air Transport Requirements Specification

Work with Boeing to develop a prototype interface for formal requirements analysis of a civil air transport was completed in 1992[40, 41]. This work, performed under a subcontract to California Polytechnic State University, included development of a Wide-Spectrum Requirements Specification Language (WSRSL) and prototype tools to support the language. Portions of a set of requirements for an Advanced Subsonic Civil Transport (ASCT) developed by a Boeing engineer under previous NASA funding were rewritten in WSRSL to demonstrate the use of the language and toolset. Since WSRSL is a formal language, the specifications can be formally analyzed for syntactic correctness, completeness, and consistency.

5 Coordination Activities

5.1 Relationship to NASA Program Offices

The formal methods research program is currently being funded by NASA Ames' Information Technology (IT) Program, Langley's Aerospace Vehicle Systems Technology (AVST) Program, Langley's Aviation Safety Program (AvSP), and Ames' Advanced Aviation Transportation Technology (AATT) Program.

5.2 FAA/RTCA Involvement

As the federal agency responsible for certification of civil air transports, the FAA shares our interest in promising approaches to engineering and validating ultrareliable flight-control systems. Additionally, because the FAA must approve any new methodologies for developing life-critical digital systems for civil air transports, their acceptance of formal methods is a necessary precursor to its adoption by industry system designers. We are working with Pete Saraceni of the FAA Technical Center to insure that our program is relevant to the certification process. The FAA has co-sponsored some of our work. John Rushby of SRI gave a tutorial on formal methods at an FAA Software Advisory Team (SWAT) meeting at their request. The SWAT team suggested that we include an assessment of formal methods in an ongoing Guidance Control Software (GCS) experiment in our branch; Odyssey Research Associates (ORA) developed a formal specification of the GCS application.

John Rushby wrote a chapter for the FAA Digital Systems Validation Handbook Volume III on formal methods[24], which is also available as a NASA contractor report [111]. The handbook provides detailed information about digital system design and validation and is used by the FAA certifiers. In preparation for this chapter, Rushby produced a comprehensive analysis of formal methods [109].

George Finelli, the former assistant Branch Head of the System Validation Methods Branch (the Branch in which the formal methods team worked before NASA Langley's reorganization in 1994) and a member of the RTCA committee formed to develop DO–178B, together with Ben Di Vito (ViGYAN Inc.), was instrumental in including formal methods as an alternate means of compliance in the DO–178B standard.

Kelly Hayhurst, a member of the formal methods team, has directed a software engineering case study of the DO-178B standard. The data from this case study is being used to train FAA certification specialists and avionics industry representatives in aspects of software certification.

Currently, members of the Langley staff are involved in RTCA committees SC-180 (Airborne Electronic Hardware), SC-182 (Minimal Operating Performance Standard for an Airborne Computer Resource), SC-190 (Committee On Application Guidelines For RTCA DO-178b/ED-12b), and in the ISO sponsored Ada Annex H Rapporteur Group (HRG).

6 Summary

The NASA Langley program in formal methods has three major goals: (1) develop formal methods technology suitable for a wide range of aerospace designs and (2) facilitate technology transfer by initiating joint projects between formal methods researchers and aerospace industries to apply the results of the research to real systems, and (3) capitalize on the formal methods technology transferred to industry to meet NASA's new goals in increasing aircraft safety and decreasing the cost of air travel.

Starting in 1991, NASA Langley initiated several aggressive projects designed to move FM into productive use in the aerospace community:

- Boeing PIU Project (1991)
- Charles Stark Draper FTPP Scoreboard Project (1991)
- Allied Signal Hybrid Fault Models (1992)
- Shuttle Tile Project (1992)
- Space Shuttle Jet Select Project (1993)
- Honeywell Navigation (1993)
- Rockwell Collins AAMP5 (1993)
- Honeywell Tablewise (1994)
- Union Switch and Signal (1994)
- Rockwell Collins AAMP-FV (1995)
- Space Shuttle GPS and 3EO upgrades (1995)
- Integrated Modular Avionics and RTCA SC-182 (1997)
- Collins Mode Confusion Project (1998)
- Translation of UML into PVS (1999)

- Formal Analysis of AILS (1999)
- SPIDER (2000)
- Honeywell Technology Center DEOS verification (2000)
- Rockwell Collins requirements analysis and mode confusion (2000)
- Honeywell Engines and Systems: FTIMA for FADEC (2000)
- Barron Associates/BF Goodrich: non-adaptive neural nets. (2000)
- Univ. of Va/Litton: Integration w/ SW lifecycle (2000)

NASA's program has advanced aerospace-related formal methods in the United States to the point where commercial exploitation of formal methods has begun in some application areas. Our program has driven the development of PVS, one of the most widely used general-purpose theorem prover in the world [97], and the Odyssey Research Associates VHDL-verification tool. Commercial industry has been anxious to work with our team, although we have not had sufficient resources to work with as many as we would have liked. Nevertheless, we have helped lay the necessary foundation for productive use of formal methods in several companies. We are now exploiting this newly developed capability in these companies to address NASA's ambitious goal of reducing the accident rate to 1/10th of today's level within 20 years.

References

[1] Saab Blames Gripen Crash on Software. *Aviation Week & Space Technology*, Feb. 1989.

[2] Barrett, Geoff: Formal Methods Applied to a Floating-Point Number System. *IEEE Transactions on Software Engineering*, vol. 15, no. 5, May 1989, pp. 611–621.

[3] Best, David W.; Charles E. Kress, Nick M. Mykris; Russell, Jeffrey D.; and Smith, William J.: An advanced-architecture CMOS/SOS microprocessor. *IEEE Micro*, vol. 2, no. 4, Aug. 1982, pp. 11–26.

[4] Bevier, William R.; and Young, William D.: *Machine Checked Proofs of the Design and Implementation of a Fault-Tolerant Circuit*. NASA Contractor Report 182099, Nov. 1990.

[5] Bevier, William R.; and Young, William D.: The Proof of Correctness of a Fault-Tolerant Circuit Design. In *Second IFIP Conference on Dependable Computing For Critical Applications*, Tucson, Arizona, Feb. 1991, pp. 107–114.

[6] Bickford, Mark; and Srivas, Mandayam: *Verification of the FtCayuga Fault-Tolerant Microprocessor System (Volume 2: Formal Specification and Correctness Theorems)*. NASA Contractor Report 187574, July 1991.

[7] Bose, Bhaskar: *DDD - A Transformation System for Digital Design Derivation*. Indiana University, Technical Report 331, Computer Science Department, May 1991.

[8] Bose, Bhaskar; and Johnson, Steven D.: DDD-FM9001: Derivation of a Verified Microprocessor. An Exercise in Integrating Verification with Formal Derivation. In Milne, G.; and Pierre, L., editors 1993:, *Proceedings of IFIP Conference on Correct Hardware Design and Verification Methods*. Springer, LNCS 683, 1993, pp. 191–202. also published as Tech Report # 380, Computer Science Department, Indiana University.

[9] Brock, Bishop; and Hunt, Jr., Warren A.: *Report on the Formal Specification and Partial Verification of the VIPER Microprocessor.* NASA Contractor Report 187540, July 1991.

[10] Butler, Ricky W.: *An Elementary Tutorial on Formal Specification and Verification Using PVS.* NASA Technical Memorandum 108991, Sept. 1993.

[11] Butler, Ricky W.; and Di Vito, Ben L.: *Formal Design and Verification of a Reliable Computing Platform For Real-Time Control (Phase 2 Results).* NASA Technical Memorandum 104196, Jan. 1992.

[12] Butler, Ricky W.; Di Vito, Ben L.; and Holloway, C. Michael: *Formal Design and Verification of a Reliable Computing Platform For Real-Time Control (Phase 3 Results).* NASA Technical Memorandum 109140, Aug. 1994.

[13] Butler, Ricky W.; and Finelli, George B.: The Infeasibility of Experimental Quantification of Life-Critical Software Reliability. In *Proceedings of the ACM SIGSOFT '91 Conference on Software for Critical Systems*, New Orleans, Louisiana, Dec. 1991, pp. 66–76.

[14] Butler, Ricky W.; and Finelli, George B.: The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software. *IEEE Transactions on Software Engineering*, vol. 19, no. 1, Jan. 1993, pp. 3–12.

[15] Butler, Ricky W.; and Johnson, Sally C.: Formal Methods For Life-Critical Software. In *Computing in Aerospace 9 Conference*, San Diego, CA, Oct. 1993, pp. 319–329.

[16] Butler, Ricky W.; Miller, Steve; Potts, Jim; and Carreno, Victor: A Formal Methods Approach to the Analysis of Mode Confusion. In *17th Digital Avionics Systems Conference (DASC'98)*, Bellevue, WA, Nov. 1998.

[17] Butler, Ricky W.; and Sjogren, Jon A.: *Hardware Proofs Using EHDM and the RSRE Verification Methodology.* NASA Technical Memorandum 100669, Dec. 1988.

[18] Butler, Ricky W., (ed.): *NASA Formal Methods Workshop 1990.* NASA Conference Publication 10052, Nov. 1990.

[19] Carreño, Victor A.: Interpretation of IEEE-854 Floating-Point Standard and Definition in the HOL system. Sept. 1995.

[20] Carreño, Victor: *The Transition Assertions Specification Method.* University of Cambridge Computer Laboratory, Technical Report 279, Jan. 1993.

[21] Carreño, Victor: Verification in Higher Order Logic of Mutual Exclusion Algorithm. In *Higher Order Logic Theorem Proving and Its Applications*, vol. 780 of *Lecture Notes in Computer Science*, pp. 502–515. Springer Verlag, Vancouver, B.C., Canada, Aug. 1993.

[22] Carreño, Victor A.; and Angellatta, Rob K.: *A Case Study for the Real-Time Experimental Evaluation of the VIPER Microprocessor.* NASA Technical Memorandum 104098, Sept. 1991.

[23] Carreño, Victor A.; and Miner, Paul S.: Specification of the IEEE-854 Floating-Point Standard in HOL and PVS. In *1995 International Workshop on Higher Order Logic Theorem Proving and its Applications*, Aspen Grove, Utah, Sept. 1995. track B paper and included in supplemental proceedings.

[24] Computer Resource Management Inc.: In *Digital Systems Validation Handbook – volume III*, no. DOT/FAA/CT-88/10. FAA.

[25] Courcoubetis, Costas, editor 1993: *Computer Aided Verification, CAV '93*, vol. 697 of *Lecture Notes in Computer Science*, Elounda, Greece, June/July 1993. Springer Verlag.

[26] Crow, Judith; and Di Vito, Ben L.: Formalizing Space Shuttle Software Requirements. In *Workshop on Formal Methods in Software Practice (FMSP '96)*, San Diego, California, Jan. 1996, pp. 40–48.

[27] Crow, Judy: *Finite-State Analysis of Space Shuttle Contingency Guidance Requirements*. NASA Contractor Report 4741, May 1996.

[28] Crow, Judy; Owre, Sam; Rushby, John; Shankar, Natarajan; and Srivas, Mandayam: A Tutorial Introduction to PVS. In *WIFT'95 Workshop on Industrial-strength Formal Specification Techniques*, Boca Raton, Florida USA, Apr. 1995.

[29] Cyrluk, David; Rajan, S.; Shankar, N.; and Srivas, M. K.: Effective Theorem Proving for Hardware Verification. In *Second International Conference on Theorem Proving in Circuit Design, Theory, Practice, and Experience*, Bad Herrenalb, Germany, Sept. 1994.

[30] Cyrluk, David A.; and Srivas, Mandayam K.: Theorem Proving: Not an Esoteric Diversion: but the Unifying Framework for Industrial Verification. In *IEEE International Conference on Computer Design (ICCD) '95*, Austin, Texas, Oct. 1995.

[31] Di Vito, Ben L.: Formalizing New Navigation Requirements for NASA's Space Shuttle. In *Formal Methods Europe (FME '96)*, Oxford, England, Mar. 1996, pp. 160–178. Lecture Notes in Computer Science 1051, Springer.

[32] Di Vito, Ben L.: *A Formal Model of Partitioning for Integrated Modular Avionics*. NASA Langley Research Center, Contractor Report 1998-208703, Hampton, VA, Aug. 1998.

[33] Di Vito, Ben L.; and Butler, Ricky W.: Provable Transient Recovery for Frame-Based, Fault-Tolerant Computing Systems. In *Real-Time Systems Symposium*, Phoenix, Az, Dec. 1992.

[34] Di Vito, Ben L.; and Butler, Ricky W.: Formal Techniques for Synchronized Fault-Tolerant Systems. In *Dependable Computing for Critical Applications 3*, Dependable Computing and Fault-Tolerant Systems, pp. 279–306. Springer Verlag, Wien New York, 1993. Also presented at 3rd IFIP Working Conference on Dependable Computing for Critical Applications, Mondello, Sicily, Italy, Sept. 14–16, 1992.

[35] Di Vito, Ben L.; Butler, Ricky W.; and Caldwell, James L.: High Level Design Proof of a Reliable Computing Platform. In *Dependable Computing for Critical Applications 2*, Dependable Computing and Fault-Tolerant Systems, pp. 279–306. Springer Verlag, Wien New York, 1992. Also presented at 2nd IFIP Working Conference on Dependable Computing for Critical Applications, Tucson, AZ, Feb. 18–20, 1991, pp. 124–136.

[36] Di Vito, Ben L.; Butler, Ricky W.; and Caldwell, James L., II: *Formal Design and Verification of a Reliable Computing Platform For Real-Time Control (Phase 1 Results)*. NASA Technical Memorandum 102716, Oct. 1990.

[37] Di Vito, Ben L.; and Roberts, Larry W.: *Using Formal Methods to Assist in the Requirements Analysis of the Space Shuttle GPS Change Request.* NASA Langley Research Center, Contractor Report 4752, Hampton, VA, Aug. 1996.

[38] Dill, David L.; Drexler, Andreas J.; Hu, Alan J.; and Yang, C. Han: Protocol Verification as a Hardware Design Aid. In *1992 IEEE International Conference on Computer Design: VLSI in COmputers and Processors*, Cambridge, MA, Oct. 1992, pp. 522–525.

[39] Eichenlaub, Carl T.; Harper, C. Douglas; and Hird, Geoffrey: *Using Penelope to Assess the Correctness of NASA Ada Software: A Demonstration of Formal Methods as a Counterpart to Testing.* NASA Contractor Report 4509, May 1993.

[40] Fisher, Gene; Frincke, Deborah; Wolber, Dave; and Cohen, Gerald C.: *Structured Representation for Requirements and Specifications.* NASA Contractor Report 187522, July 1991.

[41] Frincke, Deborah; Wolber, Dave; Fisher, Gene; and Cohen, Gerald: Requirements Specification Language (RSL) and Supporting Tools. Nov. 1992.

[42] Fura, David A.; Windley, Phillip J.; and Cohen, Gerald C.: *Formal Design Specification of a Processor Interface Unit.* NASA Contractor Report 189698, Nov. 1992.

[43] Garmen, John R.: The Bug Heard 'Round The World. *ACM Software Engineering Notes*, vol. 6, no. 5, Oct. 1981, pp. 3–10.

[44] Gibbs, W. Wayt: Software's Chronic Crisis. *Scientific American*, Sept. 1994, pp. 86–95.

[45] Goldberg, Jack; et al.: *Development and Analysis of the Software Implemented Fault-Tolerance (SIFT) Computer.* NASA Contractor Report 172146, 1984.

[46] Gong, Li; Lincoln, Patrick; and Rushby, John: Byzantine Agreement with Authentication: Observations and Applications in Tolerating Hybrid and Link Faults. In *Dependable Computing for Critical Applications (DCCA-5)*, Champaign, IL, Sept. 1995.

[47] Guaspari, David: Penelope, an Ada Verification System. In *Proceedings of Tri-Ada '89*, Pittsburgh, PA, Oct. 1989, pp. 216–224.

[48] Guaspari, David: Formally Specifying the Logic of an Automatic Guidance Controller. In *Ada-Europe Conference*, Athens, Greece, May 1991.

[49] Hamilton, Margaret: Zero-defect software: the elusive goal. *IEEE Spectrum*, Mar. 1986.

[50] Harper, Richard E.; Lala, Jay H.; and Deyst, John J.: Fault Tolerant Parallel Processor Architecture Overview. In *Proceedings of the 18th Symposium on Fault Tolerant Computing*, 1988, pp. 252–257.

[51] Hayhurst, Kelly J.; Dorsey, Cheryl A.; Knight, John C.; Leveson, Nancy G.; and McCormick, G. Frank: *Streamlining Software Aspects of Certification: Report on the SSAC Survey.* Technical report, Aug. 1999.

[52] Holloway, C. Michael: *Third NASA Formal Methods Workshop 1995.* NASA Conference Publication 10176, June 1995.

[53] Holloway, C. Michael: *Lfm2000 Fifth NASA Langley Formal Methods Workshop.* Technical Report 2000, 2000.

[54] Holloway, C. Michael; and Hayhurst, Kelly J.: *Lfm97 Fourth NASA Langley Formal Methods Workshop.* NASA Conference Publication 3356, Sept. 1997.

[55] Hoover, D. N.; Guaspari, David; and Humenn, Polar: *Applications of Formal Methods to Specification and Safety of Avionics Software.* NASA Contractor Report 4723, Apr. 1996.

[56] Hoover, Doug; and Chen, Zewei: *TBell: A Mathematical Tool for Analyzing Decision Tables.* NASA Contractor Report 195027, Nov. 1994. Note: Tbell is now known as TableWise.

[57] Hoover, Doug N.: *A Mathematical Model for Railway Control Systems.* NASA Contractor Report 198353, June 1996.

[58] Houston, Iain; and King, Steve: CICS Project Report: Experiences and Results from the Use of Z in IBM. In Prehn, S.; and Toetenel, W.J., editors 1991:, *VDM '91: Formal Software Development Methods*, Noordwijkerhout, The Netherlands, Oct. 1991, Springer Verlag, pp. 588–596. Volume 1: Conference Contributions.

[59] Hunt, Warren A.: A Formal HDL and its use in the FM9001 Verification. In Hoare, C.A.R.; and Gordon, M.J., editors 1992:, *Mechanized Reasoning in Hardware Design.* Prentice-Hall, 1992.

[60] IEEE. *IEEE Standard for Radix-Independent Floating-Point Arithmetic*, 1987. ANSI/IEEE Std 854-1987.

[61] John Kelly, et. al.: Formal Methods Demonstration Project for Space Applications - Phase I Case Study: Space Shuttle Orbit DAP Jet. Dec. 1993.

[62] Johnson, Sally C.; and Butler, Ricky W.: Design For Validation. In *AIAA/IEEE 10th Digital Avionics Systems Conference*, Los Angeles, California, Oct. 1991, pp. 487–492.

[63] Johnson, Sally C.; and Butler, Ricky W.: Design For Validation. *IEEE Aerospace and Electronics Systems*, Jan. 1992, pp. 38–43.

[64] Johnson, Sally C.; Holloway, C. Michael; and Butler, Ricky W.: *Second NASA Formal Methods Workshop 1992.* NASA Conference Publication 10110, Nov. 1992.

[65] Joyce, Ed: Software Bugs: A Matter of Life and Liability. *Datamation*, May 1987.

[66] Kalvala, Sara; Archer, Myla; and Levitt, Karl: A Methodology for Integrating Hardware Design and Verification. In *ACM International Workshop on Formal Methods in VLSI Design*, Miami, FL, Jan. 1991.

[67] Kalvala, Sara; Levitt, Karl; and Cohen, Gerald C.: *Design and Verification of a DMA Processor.* NASA contractor report, 1992. Unpublished.

[68] Knight, John C.; and Leveson, Nancy G.: An experimental evaluation of the assumptions of independence in multiversion programming. *IEEE Transactions on Software Engineering*, vol. SE-12, no. 1, Jan. 1986, pp. 96–109.

[69] Knight, John. C.; and Leveson, Nancy. G.: A Reply To the Criticisms Of The Knight & Leveson Experiment. *ACM SIGSOFT Software Engineering Notes*, Jan. 1990.

[70] Lamport, Leslie: Using Time Instead of Timeout for Fault-Tolerant Distributed Systems. *ACM Transactions on Programming Languages and Systems*, vol. 6, no. 2, Apr. 1984, pp. 254–280.

[71] Lamport, Leslie; and Melliar-Smith, P. M.: Synchronizing Clocks in the Presence of Faults. *Journal Of The ACM*, vol. 32, no. 1, Jan. 1985, pp. 52–78.

[72] Lamport, Leslie; Shostak, Robert; and Pease, Marshall: The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, July 1982, pp. 382–401.

[73] Leveson, Nancy G.: Software Safety: What, Why, and How. *Computing Surveys*, vol. 18, no. 2, June 1986.

[74] Levitt, Karl; and et. al.: *Formal Verification of a Microcoded VIPER Microprocessor using HOL.* NASA Contractor Report 4489, Feb. 1993.

[75] Lincoln, Patrick; and Rushby, John: Formal Verification of an Algorithm for Interactive Consistency under a Hybrid Fault Model. In Courcoubetis [25], pp. 292–304.

[76] Lincoln, Patrick; and Rushby, John: *A Formally Verified Algorithm For Interactice Consistency Under a Hybrid Fault Model.* NASA Contractor Report 4527, July 1993.

[77] Lincoln, Patrick; and Rushby, John: A Formally Verified Algorithm for Interactive Consistency under a Hybrid Fault Model. In *Fault Tolerant Computing Symposium 23*, Toulouse, France, June 1993, IEEE Computer Society, pp. 402–411.

[78] Lincoln, Patrick; and Rushby, John: Formal Verification of an Interactive Consistency Algorithm for the Draper FTP Architecture under a Hybrid Fault Model. In *1994 Computer Assurance (COMPASS) Conference*, June 1994.

[79] Miller, Steve; and Srivas, Mandayam: Formal Verification of the AAMP5 Microprocessor: A Case Study in the Industrial Use of Formal Methods. In *WIFT'95 Workshop on Industrial-strength Formal Specification Techniques*, Boca Raton, Florida USA, Apr. 1995, pp. 30–43.

[80] Miller, Steven P.; and Potts, James N.: *Detecting Mode Confusion Through Formal Modeling and Analysis.* Technical Report CR-1999-208971, Jan. 1999.

[81] Miner, Paul S.: *An Extension to Schneider's General Paradigm for Fault-Tolerant Clock Synchronization.* NASA Technical Memorandum 107634, Langley Research Center, Hampton, VA, 1992.

[82] Miner, Paul S.: *A Verified Design of a Fault-Tolerant Clock Synchronization Circuit: Preliminary Investigations.* NASA Technical Memorandum 107568, Mar. 1992.

[83] Miner, Paul S.: *Verification of Fault-Tolerant Clock Synchronization Systems.* NASA Technical Paper 3349, Nov. 1993.

[84] Miner, Paul S.: *Defining the IEEE-854 Floating-Point Standard in PVS.* NASA, NASA Technical Memorandum 110167, Langley Research Center, Hampton, VA, June 1995.

[85] Miner, Paul S.: *Hardware Verification using Coinductive Assertions.* PhD thesis, Computer Science Department, Indiana University, USA, 1997.

[86] Miner, Paul S.; and Johnson, Steven D.: Verification of an Optimized Fault-Tolerant Clock Synchronization Circuit. In Sheeran, Mary; and Singh, Satnam, editors 1996:, *Designing Correct Circuits*, Electronic Workshops in Computing, Bastad, Sweden, Sept. 1996, Springer-Verlag.

[87] Miner, Paul S.; and Leathrum, James F., Jr.: Verification of IEEE Compliant Subtractive Division Algorithms. In Srivas, Mandayam K., editor 1996:, *Formal Methods in Computer-Aided Design, FMCAD '96*, Lecture Notes in Computer Science, Palo Alto, CA, Nov. 1996, Springer-Verlag. To Appear.

[88] Miner, Paul S.; Padilla, Peter A.; and Torres, Wilfredo: A Provably Correct Design of a Fault-Tolerant Clock Synchronization Circuit. In *11th Digital Avionics Systems Conference*, Seattle, WA, Oct. 1992, pp. 341–346.

[89] Miner, Paul S.; Pullela, Shyamsundar; and Johnson, Steven D.: Interaction of Formal Design Systems in the Development of a Fault-Tolerant Clock Synchronization Circuit. In *13th Symposium on Reliable Distributed Systems.* IEEE Computer Society Press, 1994, pp. 128–137. Proceedings of SRDS 94 held at Dana Point, California, October 1994.

[90] Moore, J Strother: *A Formal Model of Asynchronous Communication and Its Use in Mechanically Verifying a Biphase Mark Protocol.* NASA Contractor Report 4433, June 1992.

[91] Moore, J Strother: *Mechanically Verified Hardware Implementing an 8-bit Parallel IO Byzantine Agreement Processor.* NASA Contractor Report 189588, Apr. 1992.

[92] National Aeronautics and Space Administration, Office of Safety and Mission Assurance, Washington, DC. *Formal Methods Specification and Verification Guidebook for Software and Computer Systems, Volume I: Planning and Technology Insertion*, July 1995.

[93] National Aeronautics and Space Administration, Office of Safety and Mission Assurance, Washington, DC. *Formal Methods Specification and Verification Guidebook for Software and Computer Systems, Volume II: A Practitioner's Companion*, May 1997.

[94] Neumann, Peter G.: Some Computer-Related Disasters and Other Egregious Horrors. *ACM Software Engineering Notes*, vol. 10, no. 1, Jan. 1985, pp. 6–12.

[95] Owre, S.; Shankar, N.; and Rushby, J. M.: *The PVS Specification Language (Beta Release).* Computer Science Laboratory, SRI International, Menlo Park, CA, Feb. 1993.

[96] Owre, S.; Shankar, N.; and Rushby, J. M.: *User Guide for the PVS Specification and Verification System (Beta Release).* Computer Science Laboratory, SRI International, Menlo Park, CA, Feb. 1993.

[97] Owre, Sam; Rushby, John; ; Shankar, Natarajan; and von Henke, Friedrich: Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS. *IEEE Transactions on Software Engineering*, vol. 21, no. 2, Feb. 1995, pp. 107–125.

[98] Owre, Sam; Rushby, John; and Shankar, Natarajan: *Analyzing Tabular and State-Transition Requirements Specifications in PVS.* NASA Contractor Report 201729, July 1997.

[99] Owre, Sam; Rushby, John; Shankar, Natarajan; and Srivas, Mandayam: A Tutorial Using PVS For Hardware Verification. In *Second International Conference on Theorem Proving in Circuit Design, Theory, Practice, and Experience*, Bad Herrenalb, Germany, Sept. 1994.

[100] Owre, Same; and Shankar, Natarajan: *Abstract Datatypes in PVS*. NASA Contractor Report 97-206264, Nov. 1997.

[101] Pan, Jing; and Levitt, Karl: Towards a Formal Specification of the IEEE Floating-Point Standard with Application to the Verification of Floating-Point Coprocessors. In *24th Asilomar Conference on Signals, Systems & Computers*, Monterrey, CA., Nov. 1990.

[102] Pan, Jing; Levitt, Karl; and Cohen, Gerald C.: *Toward a Formal Verification of a Floating-Point Coprocessor and its Composition with a Central Processing Unit*. NASA Contractor Report 187547, Aug. 1991.

[103] Pan, Jing; Levitt, Karl; and Schubert, E. Thomas: Toward a Formal Verification of a Floating-Point Coprocessor and its Composition with a Central Processing Unit. In *ACM International Workshop on Formal Methods in VLSI Design*, Miami, FL, Jan. 1991.

[104] Rajan, S.; Shankar, N.; and Srivas, M. K.: An Integration of Model Checking with Automated Proof Checking. In *Computer Aided Verification (CAV 95)*, Liege, Belgium, July 1995.

[105] Roberts, Larry W.; and Beims, Mike: *Using Formal Methods to Assist in the Requirements Analysis of the Space Shuttle HAC Change Request (CR 90960E)*. NASA Johnson Space Center, Technical report, 1996. To appear.

[106] Rogers, Michael; and Gonzalez, David L.: Can We Trust Our Software? *Newsweek*, Jan. 1990.

[107] Rushby, John: *Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems*. NASA Contractor Report 4384, July 1991.

[108] Rushby, John: *Formal verification of an Oral Messages algorithm for interactive consistency*. NASA Contractor Report 189704, Oct. 1992.

[109] Rushby, John: *Formal Methods and Digital Systems Validation for Airborne Systems*. NASA Contractor Report 4551, Dec. 1993.

[110] Rushby, John: A Formally Verified Algorithm Clock Sychronization Under a Hybrid Fault Model. In *ACM Principles of Distributed Computing '94*, Aug. 1994.

[111] Rushby, John: *Formal Methods and Their Role in Digital Systems Validation for Airborne Systems*. NASA Contractor Report 4673, Aug. 1995.

[112] Rushby, John: Reconfiguration and Transient Recovery in State-Machine Architectures. In *26th Annual International Symposium on Fault-tolerant Computing (FTCS-26)*, Sendai, Japan, June 1996.

[113] Rushby, John: Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms. In Meadows, Catherine; and Sanders, William, editors 1997:, *Dependable Computing for Critical Applications—6*, Garmisch-Partenkirchen, Germany, Mar. 1997, IEEE Computer Society, pp. 191–210.

[114] Rushby, John: *Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance*. NASA Contractor Report CR-1999-209347, June 1999.

[115] Rushby, John; and von Henke, Friedrich: *Formal Verification of a Fault-Tolerant Clock Synchronization Algorithm*. NASA Contractor Report 4239, June 1989.

[116] Rushby, John; and von Henke, Friedrich: Formal Verification of Algorithms for Critical Systems. *IEEE Transactions on Software Engineering*, vol. 19, no. 1, Jan. 1993, pp. 13–23.

[117] Schneider, Fred B.: *Understanding Protocols for Byzantine Clock Synchronization*. Cornell University, Ithaca, NY, Technical Report 87-859, Aug. 1987.

[118] Schrage, Michael: 'When the Chips Are Down' Will Likely Be Heard More Often in Computing. *The Washington Post*, pp. B3. December 16, 1994.

[119] Schubert, Thomas; and Levitt, Karl: Verification of Memory Management Units. In *Second IFIP Conference on Dependable Computing For Critical Applications*, Tucson, Arizona, Feb. 1991, pp. 115–123.

[120] Schubert, Thomas; Levitt, Karl; and Cohen, Gerald C.: *Towards Composition of Verified Hardware Devices*. NASA Contractor Report 187504, Nov. 1991.

[121] Schubert, Thomas; Levitt, Karl; and Cohen, Gerald C.: *Formal Mechanization of Device Interactions With a Process Algebra*. NASA Contractor Report 189644, Nov. 1992.

[122] Schubert, Thomas; Levitt, Karl; and Cohen, Gerald C.: *Formal Verification of a Set of Memory Management Units*. NASA Contractor Report 189566, 1992.

[123] Shankar, N.; and Owre, S.: *PVS Semantics*. NASA Contractor Report yyy, 1998.

[124] Shankar, N.; Owre, S.; and Rushby, J. M.: *The PVS Proof Checker: A Reference Manual (Beta Release)*. Computer Science Laboratory, SRI International, Menlo Park, CA, Feb. 1993.

[125] Shankar, Natarajan: *Mechanical Verification of a Schematic Byzantine Clock Synchronization Algorithm*. NASA Contractor Report 4386, July 1991.

[126] Shankar, Natarajan: Mechanical Verification of a Generalized Protocol for Byzantine Fault-Tolerant Clock Synchronization. In *Second International Symposium on Formal Techniques in Real Time and Fault Tolerant Systems*, vol. 571 of *Lecture Notes in Computer Science*, pp. 217–236. Springer Verlag, Nijmegen, The Netherlands, Jan. 1992.

[127] Shankar, Natarajan: Verification of Real-Time Systems Using PVS. In Courcoubetis [25], pp. 280–291.

[128] Shankar, Natarajan; Owre, Sam; and Rushby, John: *PVS Tutorial*. Computer Science Laboratory, SRI International, Menlo Park, CA, Feb. 1993. Also appears in Tutorial Notes, *Formal Methods Europe '93: Industrial-Strength Formal Methods*, pages 357–406, Odense, Denmark, April 1993.

[129] Srivas, Mandayam: *Efficient Validation of Superscalar Microprocessors*. NASA contractor report, 1998. To be published.

[130] Srivas, Mandayam; and Bickford, Mark: *Verification of the FtCayuga Fault-Tolerant Microprocessor System (Volume 1: A Case Study in Theorem Prover-Based Verification).* NASA Contractor Report 4381, July 1991.

[131] Srivas, Mandayam; and Bickford, Mark: *Moving Formal Methods Into Practice: Verifying the FTPP Scoreboard: Phase 1 Results.* NASA Contractor Report 189607, May 1992.

[132] Srivas, Mandayam; and Miller, Steve: *Formal Verification of an Avionics Microprocessor.* NASA Contractor Report 4682, July 1995.

[133] Walter, C. J.; Kieckhafer, R. M.; and Finn, A. M.: MAFT: A Multicomputer Architecture for Fault-Tolerance in Real-Time Control Systems. In *Real Time Systems Symposium*, Dec. 1985.

[134] Wiener, Lauren Ruth: *Digital Woes.* Addison-Wesley Publishing Company, 1993. ISBN 0-201-62609-8.

[135] Windley, Phillip J.: Abstract Hardware. In *ACM International Workshop on Formal Methods in VLSI Design*, Miami, FL, Jan. 1991.

[136] Windley, Phillip J.: The Formal Verification of Generic Interpreters. In *28th Design Automation Conference*, San Franciso, CA, June 1991.

[137] Windley, Phillip J.; Levitt, Karl; and Cohen, Gerald C.: *Formal Proof of the AVM-1 Microprocessor Using the Concept of Generic Interpreters.* NASA Contractor Report 187491, Mar. 1991.

[138] Windley, Phillip J.; Levitt, Karl; and Cohen, Gerald C.: *The Formal Verification of Generic Interpreters.* NASA Contractor Report 4403, Oct. 1991.

[139] Young, William D.: *Verifying the Interactive Convergence Clock Synchronization Algorithm Using the Boyer-Moore Theorem Prover.* NASA Contractor Report 189649, Apr. 1992.